Selection Sort

Introduction

Selection sort divides the input list into two parts: a sorted sublist of items which is built up from left to right at the front (left) of the list and a sublist of the remaining unsorted items that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list.
The algorithm proceeds by finding the smallest (or largest) element in the unsorted sublist, swapping it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right [4].

Performance

In general, the performance of the sorting algorithm is analyzed from two aspects, time complexity and space complexity. Time complexity represents the time taken to execute the algorithm, which is generally considered in three cases: best case, worst case, and average case [6]. Space complexity represents the amount of memory required to complete a program [5].

Use **array** as data structure and **n** denotes the input array size, selection sort performance is as follows [4]:

Worst-case time complexity	$\boldsymbol{O(n^2)}$
Average time complexity	$\boldsymbol{O(n^2)}$
Best-case time complexity	$\boldsymbol{O(n^2)}$
Worst-case space complexity	$\boldsymbol{O(1)}$

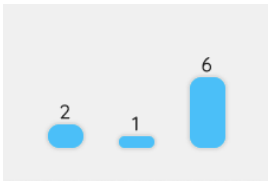
Pseudocode [3]

Algorithm: SelectionSort(Arr)
Input: an array of integers Arr
Output: The result of sorting Arr

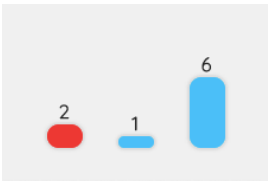
```
n = Arr.length
for j=1 to n-1 do
  smallest = j
  for i = j+1 to n do
    if seq[i] < seq[smallest] then
      smallest = i
    end if
  end for
  swap(Arr[j],Arr[smallest])
end for

return Arr
```

Example

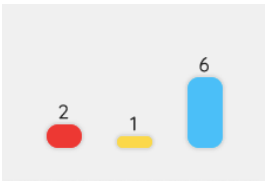
Sorting the list: 2,1,6



(1)



(2)

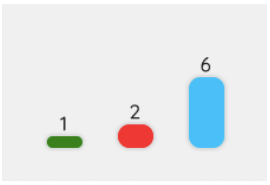


(3)

(4)

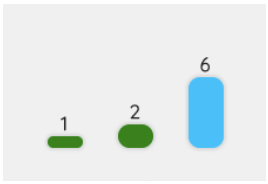
(5)

(6)



(7)

(8)



(9)

(10)

Implement in programming language

Java [1]

```java
public void selectionSort(int arr[])
    {
        int n = arr.length;

        for (int i = 0; i < n-1; i++)
        {
            int min_idx = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;

            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }
```

JavaScript [2]

```javascript
function selectionSort(arr) {
    var len = arr.length;
    var minIndex, temp;
    for (var i = 0; i < len - 1; i++) {
        minIndex = i;
        for (var j = i + 1; j < len; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
    return arr;
}
```

C [2]

```c
void swap(int *a,int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
void selection_sort(int arr[], int len)
{
    int i,j;

        for (i = 0 ; i < len - 1 ; i++)
    {
                int min = i;
                for (j = i + 1; j < len; j++)
                        if (arr[j] < arr[min])
                                min = j;
                swap(&arr[min], &arr[i]);
        }
}
```

References

1. GeeksforGeeks Contributors (2018). *Java Program for Selection Sort.* [Online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/java-program-for-selection-sort/ [Accessed 6 Mar. 2021].
2. Runoob Contributors (2018). *Selection sort*. [Online] Runoob. Available at: https://www.runoob.com/w3cnote/selection-sort.html [Accessed 6 Mar. 2021].
3. Visualgo Contributors (2014). [Software] Visualgo. Available at: https://visualgo.net/en/sorting [Accessed 6 Mar. 2021].
4. Wikipedia Contributors (2021). *Selection sort*. [Online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Selection_sort [Accessed 6 Mar. 2021].
5. Wikipedia Contributors (2021). *Space complexity*. [Online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Space_complexity [Accessed 20 Mar. 2021].
6. Wikipedia Contributors (2021). *Time complexity*. [Online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Time_complexity [Accessed 20 Mar. 2021].